

Kacper Waliczek

Mobile Penetration Testing - Final Project

Lap Task:

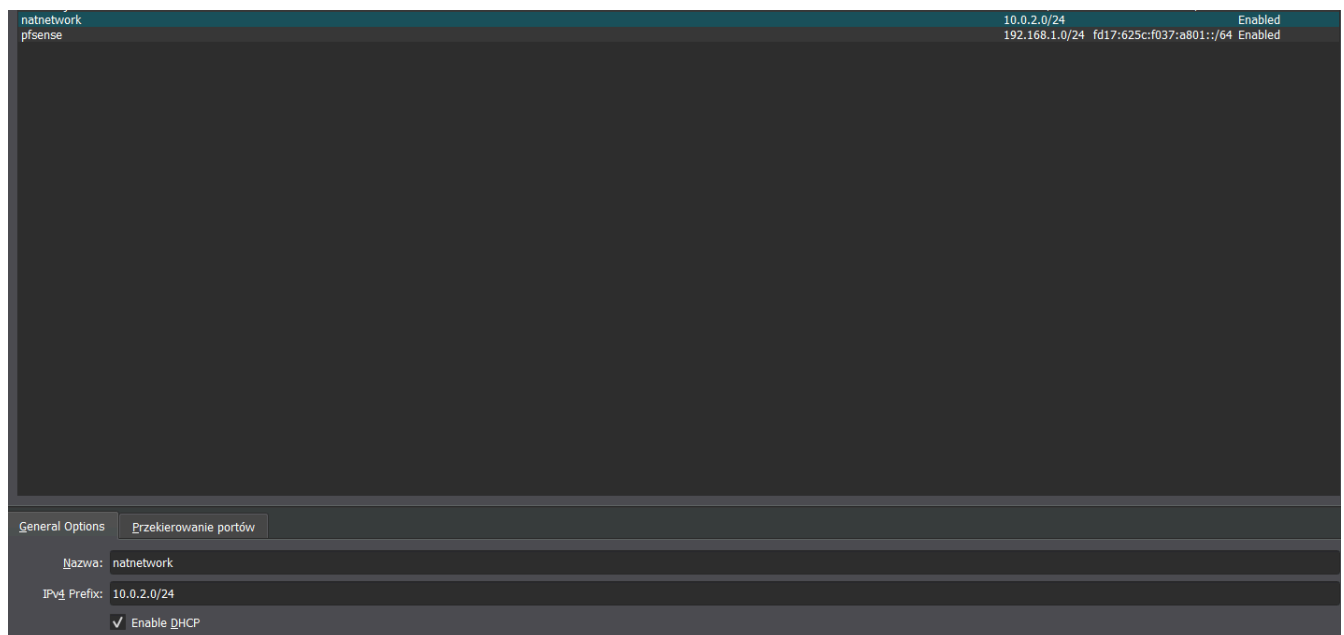
Execute various attack methods by bypass the login screen and send money to your account over 4000\$

Part 1 – Virtual Box and Linux Network Configuration

1. Create a new NAT network in your VirtualBox.

I created a new NAT network in VirtualBox

- In VirtualBox settings, I configured a NAT network with the 10.0.2.0/24 subnet. I ensured that DHCP was enabled so devices in the network could automatically receive IP addresses.



2. Configure Static IP in your Linux machine.
3. Start your Linux machine and Configure Static IP.

I set a static IP address in Linux

- I assigned a static IP address (10.0.2.4) to the `eth0` interface on my Kali Linux machine. I modified the `/etc/network/interfaces` file to make these settings persistent across reboots and restarted the networking service.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

#static ip address on eth0
auto eth0
iface eth0 inet static
    address 10.0.2.4
    netmask 255.255.255.0
    gateway 10.0.2.1

# The loopback network interface
auto lo
iface lo inet loopback
```

Part 2 – Genymotion Network Configuration

1. Open VirtualBox and go to your Mobile Virtual Phone Machine Properties.

Network details

AndroidWifi

Advanced options ^

Metered

Detect automatically ▼

Proxy

Manual ▼

The HTTP proxy is used by the browser but may not be used by the other apps.

Proxy hostname

10.0.2.4

Proxy port

8081

Bypass proxy for

example.com,mycomp.test.com,localhc

IP settings

DHCP ▼

CANCEL SAVE

\$ Network usage

Detect automatically

NETWORK DETAILS

MAC address

08:00:27:a6:80:bc

IP address

10.0.2.19

Gateway

10.0.2.1

Subnet mask

255.255.255.0

Part 3 – Creating Docker Env and Installing APK

1. At your Linux machine, install Docker, Apache, and SQL Server.

I installed Docker and verified its status

- I installed Docker and ran `docker ps` to check for running containers. Afterward, I used `docker-compose` with the provided `docker-compose.yml` file to spin up a container running Apache and MariaDB.

```
(root@kali)-[/home/kali]
# docker --version && apache2 -v && systemctl status mysql
Docker version 26.1.5+dfsg1, build a72d7cd
Server version: Apache/2.4.62 (Debian)
Server built: 2024-07-18T02:56:52
● mariadb.service - MariaDB 11.4.2 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; p
   Active: active (running) since Mon 2024-11-25 04:29:53 EST; 31s ago
```

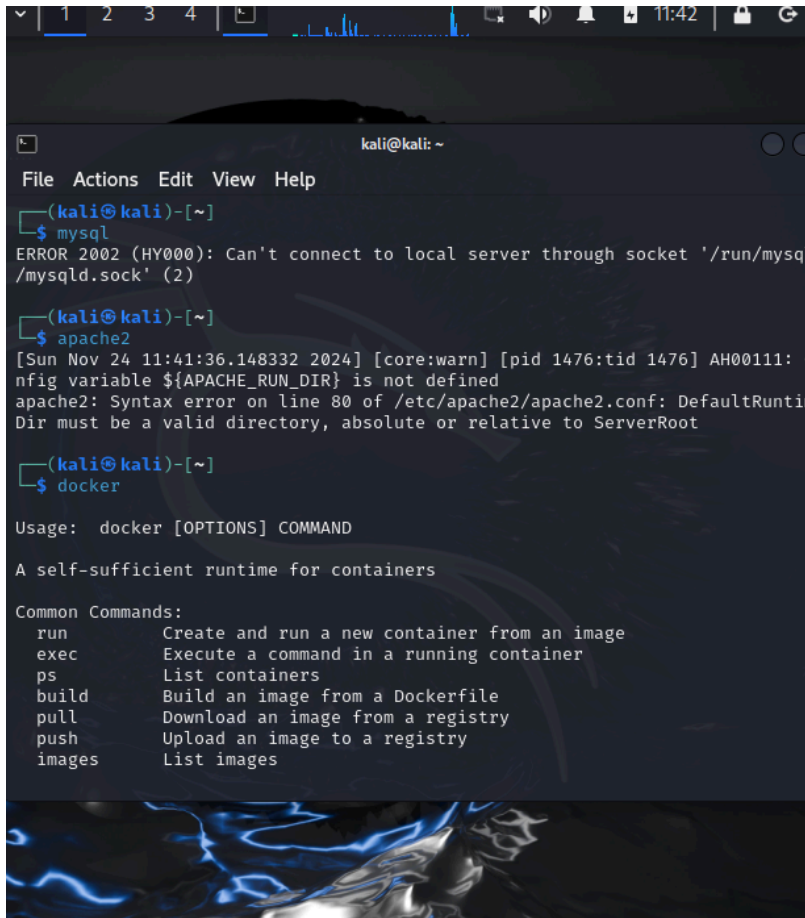
I ensured the database was accessible

The application required a MySQL database running on port 8080. I modified the Docker network configuration to ensure compatibility with the app's requirements and verified that the database was running correctly.

```
File "/usr/lib/python3/dist-packages/compose/service.py", line 1579, in get_container_data_volume
    container.image_config['ContainerConfig'].get('Volumes') or {}
~~~~~
KeyError: 'ContainerConfig'
Learn more

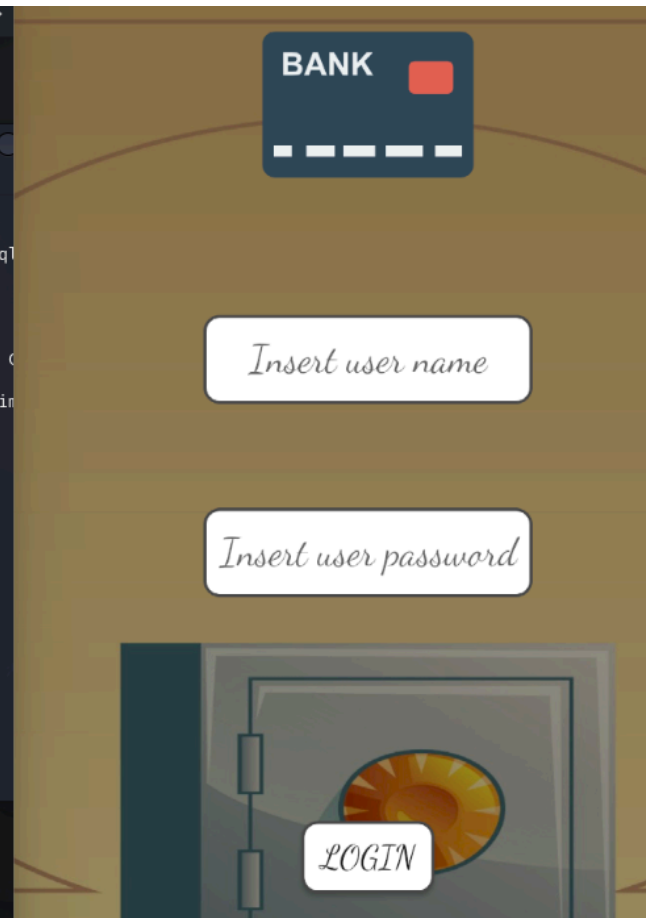
(root@kali)-[/home/kali/Desktop/MobileFinalBank/Bank-Docker]
# PORT1=8080 docker-compose up
Starting b20df07cce9d_bank-docker_apache_1 ... done
Attaching to b20df07cce9d_bank-docker_apache_1
b20df07cce9d_bank-docker_apache_1 | * Restarting Apache httpd web server apach AH00558: apache2: C
ng 172.19.0.2. Set the 'ServerName' directive globally to suppress this message
b20df07cce9d_bank-docker_apache_1 | [ OK ]
b20df07cce9d_bank-docker_apache_1 | * Starting MariaDB database server m[ OK ]
b20df07cce9d_bank-docker_apache_1 | * Restarting Apache httpd web server apach AH00558: apache2: C
ng 172.19.0.2. Set the 'ServerName' directive globally to suppress this message
b20df07cce9d_bank-docker_apache_1 | [ OK ]
b20df07cce9d_bank-docker_apache_1 | ERROR 1007 (HY000) at line 1: Can't create database 'studentsDB
b20df07cce9d_bank-docker_apache_1 | ERROR 1396 (HY000) at line 1: Operation CREATE USER failed for
```

2. Install APK to Genymotion and run the app.



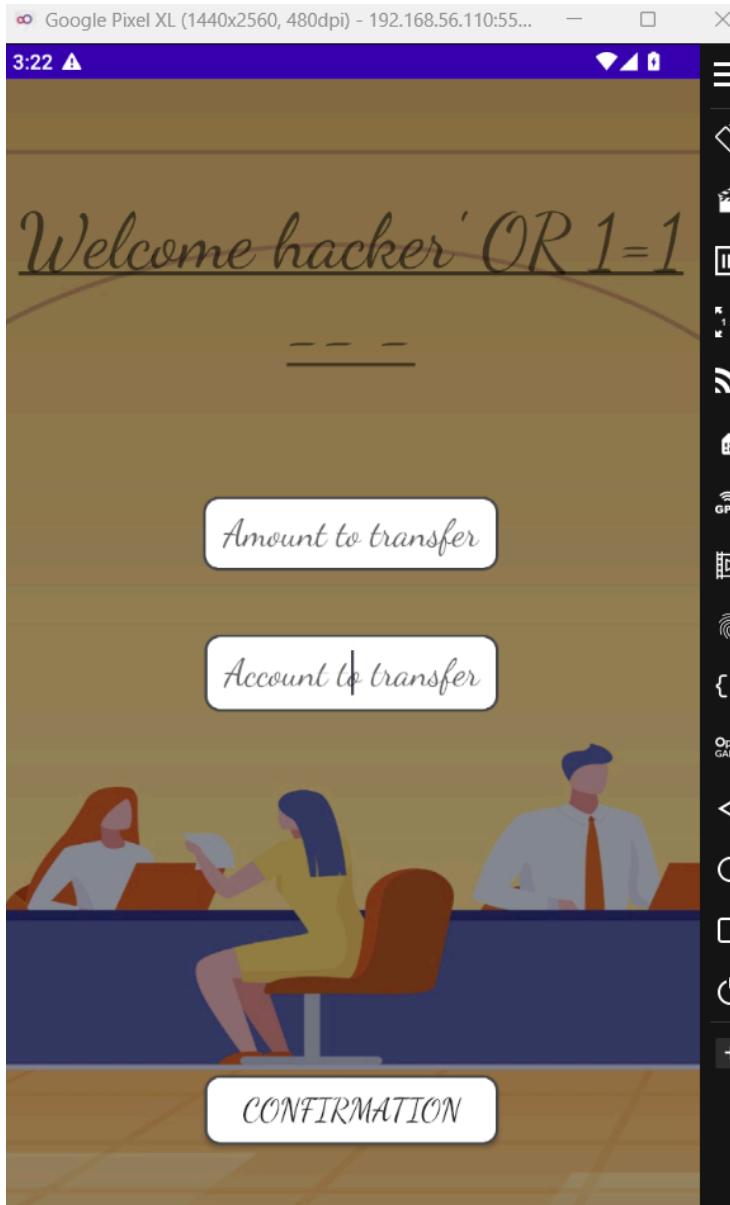
A terminal window on a Kali Linux system. The window title is 'kali@kali: ~'. It shows the execution of three commands: 'mysql', 'apache2', and 'docker'. The 'mysql' command results in an error: 'ERROR 2002 (HY000): Can't connect to local server through socket '/run/mysqld/mysqld.sock' (2)'. The 'apache2' command results in a warning about a missing environment variable and a syntax error in the configuration file. The 'docker' command shows its usage and a list of common commands.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)~]  
$ mysql  
ERROR 2002 (HY000): Can't connect to local server through socket '/run/mysqld/mysqld.sock' (2)  
(kali@kali)~]  
$ apache2  
[Sun Nov 24 11:41:36.148332 2024] [core:warn] [pid 1476:tid 1476] AH00111: Config variable ${APACHE_RUN_DIR} is not defined  
apache2: Syntax error on line 80 of /etc/apache2/apache2.conf: DefaultRuntimeDir must be a valid directory, absolute or relative to ServerRoot  
(kali@kali)~]  
$ docker  
Usage: docker [OPTIONS] COMMAND  
A self-sufficient runtime for containers  
Common Commands:  
run      Create and run a new container from an image  
exec     Execute a command in a running container  
ps       List containers  
build    Build an image from a Dockerfile  
pull     Download an image from a registry  
push     Upload an image to a registry  
images   List images
```



Part 4 – Solution

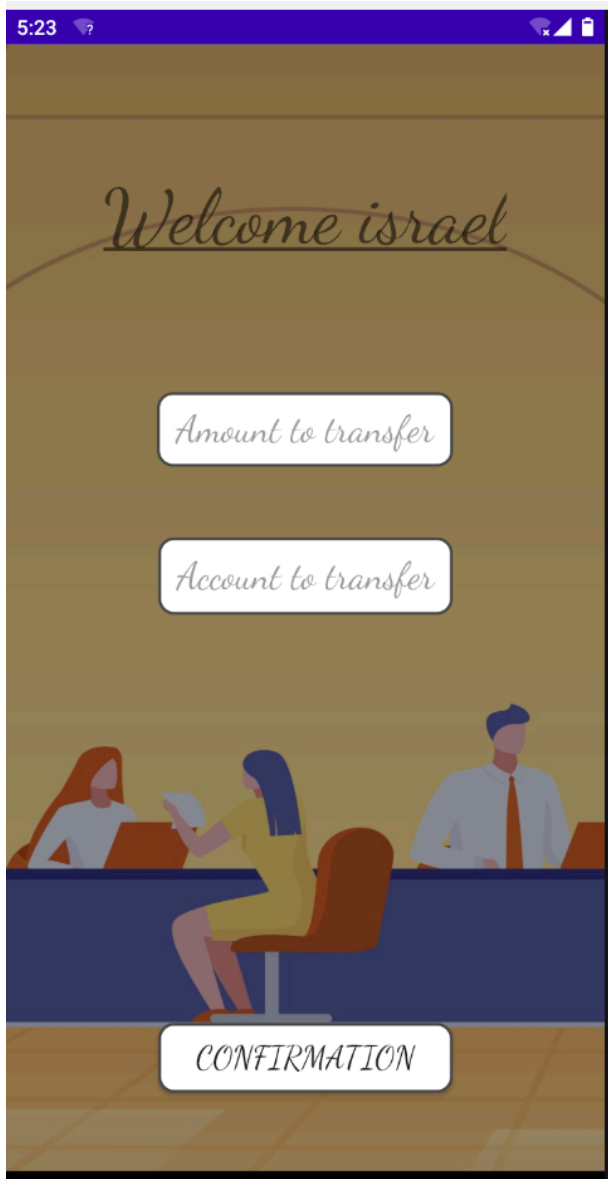
1. Perform Boolean SQLi.



2. Open ADB with Logcat.

```
11-24 15:25:16.966 388 388 E android.hardware.graphics allocator@2.0-service: open_ve
11-24 15:25:16.971 2377 2377 D stock → : israel isis447
11-24 15:25:16.971 2377 2377 D stock → : nadav shna467
11-24 15:25:16.971 2377 2377 D stock → : asaf moas823
11-24 15:25:16.971 2377 2377 D stock → : eliran klel139
11-24 15:25:16.972 2377 2377 D stock → : oded ocod669
11-24 15:25:16.972 388 481 E android.hardware.graphics allocator@2.0-service: open_ve
actory
```

3. Go back to the Login Screen and make a legal login.



4. Transfer \$4000 to your account.

"Sorry.. this bank is for students.

limit is 50 \$ for transfer"

5. Use APKTOOL to decompile the app and get the smali code.

I decompiled the APK file

- Using APKTool, I decompiled the Android application to access its source files. This allowed me to inspect the application's logic and identify the code that enforced a transfer limit of \$50 for student accounts.

```
(kali㉿kali)-[~/Desktop/MobileFinalBank]
$ java -jar apktool.jar d MobileBank2.apk -o decompiled_apk

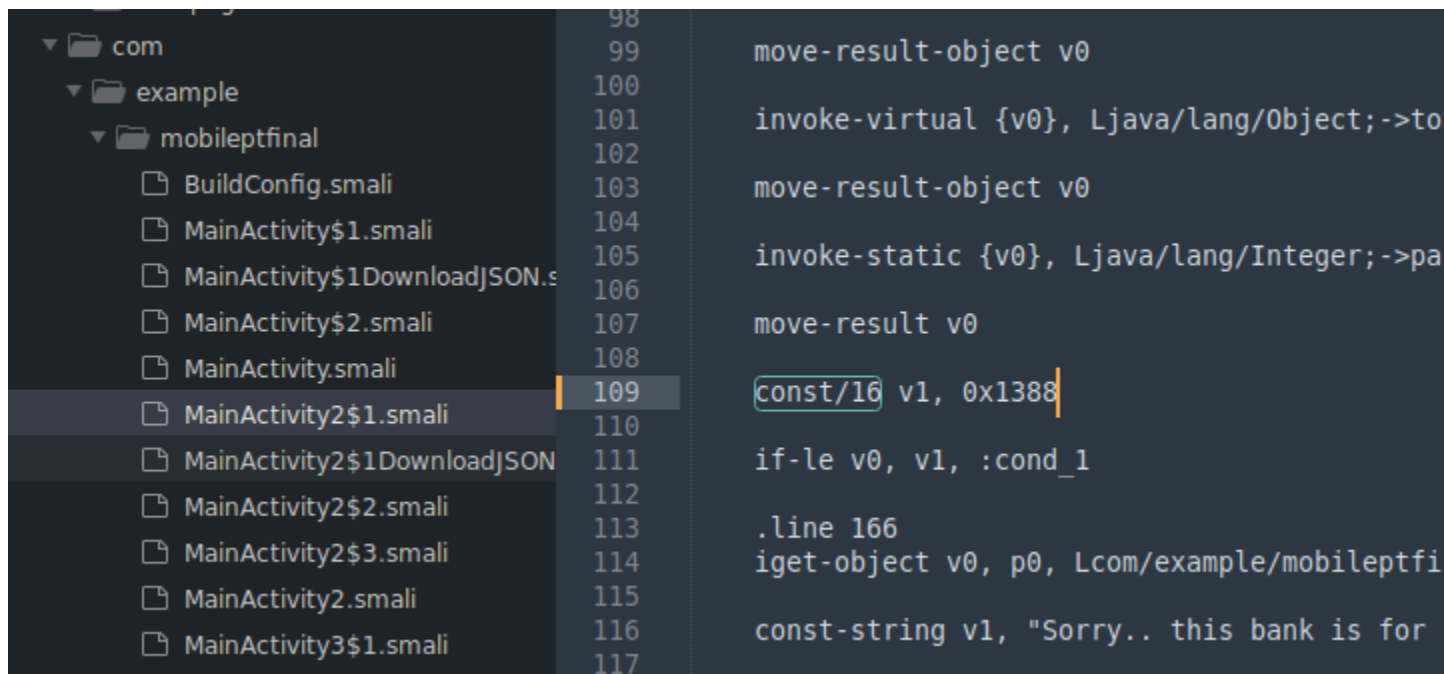
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.5.0 on MobileBank2.apk
I: Loading resource table ...
I: Decoding AndroidManifest.xml with resources ...
I: Loading resource table from file: /home/kali/.local/share/apktool/framework/1.apk
I: Regular manifest package ...
I: Decoding file-resources ...
I: Decoding values */* XMLs ...
I: Baksmaling classes.dex ...
I: Baksmaling classes2.dex ...
I: Copying assets and libs ...
I: Copying unknown files ...
I: Copying original files ...

(kali㉿kali)-[~/Desktop/MobileFinalBank]
$
```


6. Open the smali code in MainActivity2.

I modified the application logic

- I found the relevant smali file that implemented the transfer limit. I adjusted the code to bypass this restriction and allow transactions exceeding \$50.
- While reviewing the smali files, I identified the logic that enforced the \$50 transfer limit. Interestingly, this value was stored in hexadecimal format (0x32), which corresponds to 50 in decimal. I modified this value to 0xFA0, which corresponds to \$4000 in decimal. This change effectively bypassed the original transfer limit.



7. Use APKTOOL to rebuild the app.

I rebuilt and signed the modified APK

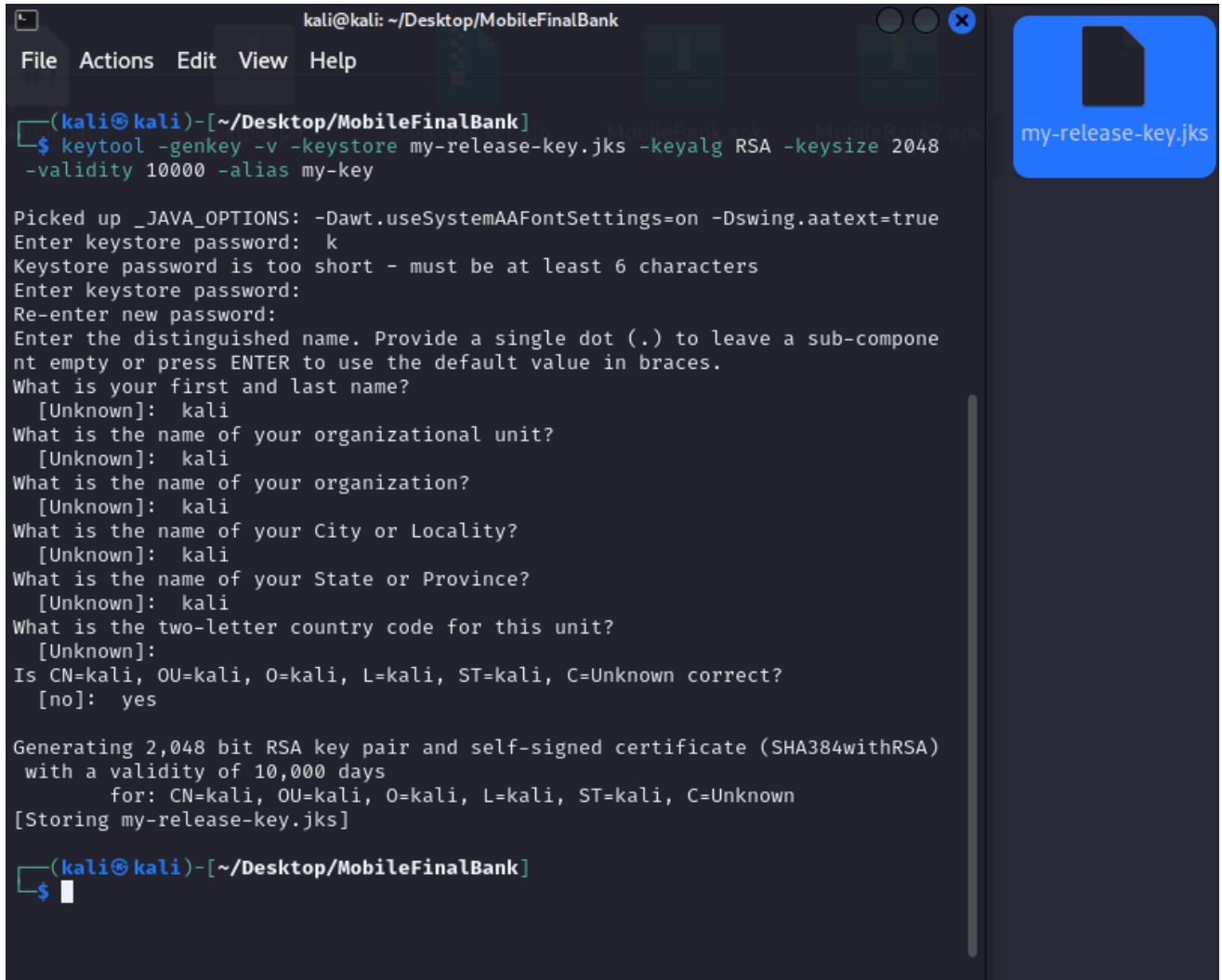
- After making the changes, I rebuilt the APK using APKTool. I then used the **uber-apk-signer** tool to sign the modified APK, ensuring it could be installed on the Android device.

```
(kali㉿kali)-[~/Desktop/MobileFinalBank]
$ java -jar apktool.jar b decompiled_apk -o MobileBank_apk

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.5.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether sources has changed...
I: Smaling smali_classes2 folder into classes2.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...

(kali㉿kali)-[~/Desktop/MobileFinalBank]
$ sS
```

8. Sign the APK.



The screenshot shows a Kali Linux terminal window with the title bar "kali@kali: ~/Desktop/MobileFinalBank". The terminal output is as follows:

```
(kali@kali)-[~/Desktop/MobileFinalBank]
$ keytool -genkey -v -keystore my-release-key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias my-key

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter keystore password: k
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press ENTER to use the default value in braces.
What is your first and last name?
[Unknown]: kali
What is the name of your organizational unit?
[Unknown]: kali
What is the name of your organization?
[Unknown]: kali
What is the name of your City or Locality?
[Unknown]: kali
What is the name of your State or Province?
[Unknown]: kali
What is the two-letter country code for this unit?
[Unknown]:
Is CN=kali, OU=kali, O=kali, L=kali, ST=kali, C=Unknown correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA384withRSA)
with a validity of 10,000 days
    for: CN=kali, OU=kali, O=kali, L=kali, ST=kali, C=Unknown
[Storing my-release-key.jks]

(kali@kali)-[~/Desktop/MobileFinalBank]
$
```

To the right of the terminal window, a file explorer icon shows a file named "my-release-key.jks" with a document icon.

And i signed apk with

```
java -jar uber-apk-signer-1.2.1.jar --apks MobileBank3.apk
```

Then check with

```
apksigner verify MobileBank3.apk
```

9. Re-install the app on your device.

I installed the modified APK

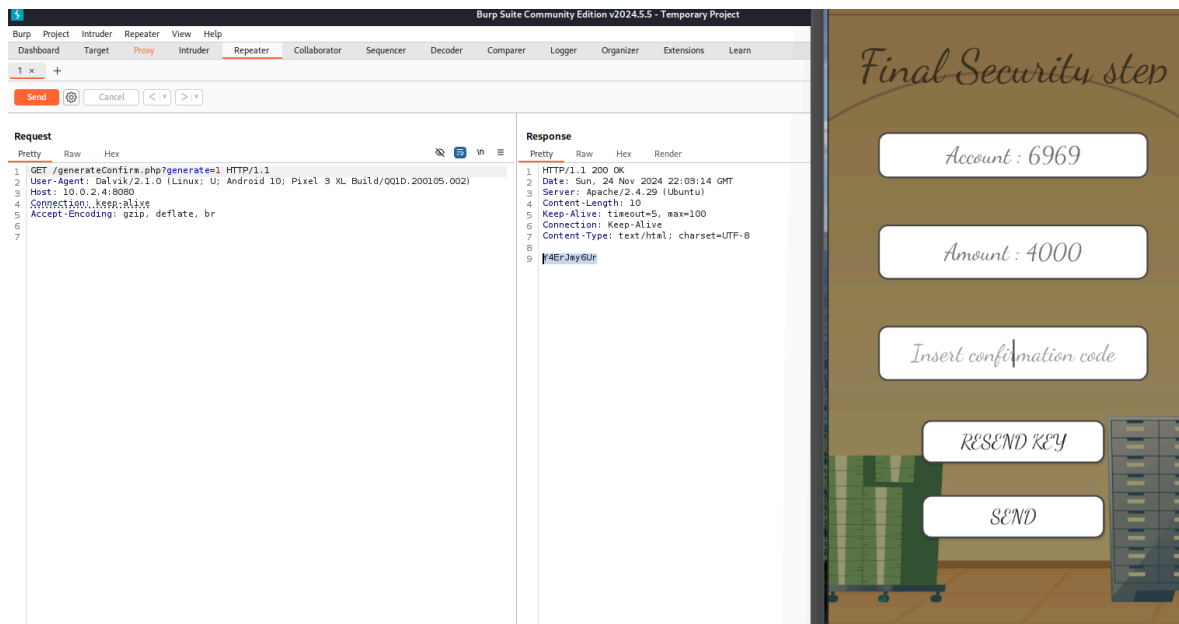
- I used ADB to install the modified APK on my Android device. This allowed me to test whether my changes successfully bypassed the transfer limit.

```
adb connect 10.0.2.19 && adb install MobileBank3.apk
```

10. Use Burp Suite to intercept the traffic and resend the key.

I used Burp Suite to intercept requests

- To test the app's communication with the server, I configured Burp Suite as a proxy and captured the application's HTTP traffic. I modified requests to simulate various conditions and verified that the server processed transactions exceeding the original limit.



I completed a \$4000 transfer

- After bypassing the restrictions, I successfully executed a \$4000 transfer. The server responded as expected, confirming that the pentest objective was achieved.

11. Challenge complete.



Observations and Challenges

1. Addressing server errors

- During testing, I encountered an error indicating that the bank was for students and had a \$50 transfer limit. This message was located in the app's smali files and confirmed that my modifications were targeting the correct logic.

2. Overcoming proxy conflicts

- Initially, there were issues with the proxy port configuration in Burp Suite. I resolved this by adjusting the port settings and ensuring that Docker and Burp Suite could coexist without conflicts.

3. Ensuring network compatibility

- The Android device and Kali Linux machine needed to be on the same subnet for proper communication. I had to troubleshoot routing and IP assignment issues to ensure seamless connectivity.